PUC-Rio
Departamento de Informática
Prof. Marcus Vinicius S. Poggi de Aragão
Horário: 2as-feiras e 4as-feiras de 9 às 11 horas - Sala 546L
25 de junho de 2008
Data da Entrega: 10 de julho de 2008
Período: 2008.1

ANÁLISE DE ALGORITMOS (INF 1721)

2^o Trabalho de Implementação

Descrição

Este trabalho prático consiste em desenvolver um algoritmo de enumeração (implícita) e o código correspondente. O objetivo é verificar as limitações computacionais na resolução de problemas onde algoritmos com complexidade menor são desconhecidos. Para isso o desempenho das implementações destes algoritmos deverá ser analisado com respeito ao tempo de CPU.

O desenvolvimento do códigos e a análise deve seguir o seguinte roteiro:

- Descrever os algoritmos informalmente.
- Demonstrar o entendimento do algoritmo explicando, em detalhe, o resultado que o algoritmo deve obter e justificá-lo.
- Explicar a fundamentação do algoritmo e justificar a sua corretude. Apresentar e explicar a complexidade teórica esperada para cada algoritmo.
- Documente o arquivo contendo o código fonte de modo que cada passo do algoritmo esteja devidamente identificado e deixe claro como este passo é executado.

A corretude código será testada sobre os conjuntos de instâncias distribuidos. O trabalho entregue deve conter:

- Um documento contendo o roteiro de desenvolvimento dos algoritmos (e dos códigos), os itens pedidos acima, comentários e análises sobre a implementação e os testes realizados (papel). **É parte importante do documento uma tabela contendo os tempos de CPU obtidos para cada instância dada**.
- Um e-mail para poggi@inf.puc-rio.br deve ser enviado anexando um arquivo **zip** contendo os códigos fonte, os executáveis correspondentes e o documento final (é obrigatório o uso do ASSUNTO (ou SUBJECT) AA062T2). (Lembrem que o gmail não aceita .exe, e portanto vc deve modificar a terminação de .zip para .txt).
- O trabalho pode ser feito em grupo de até 2 alunos.

1. O problema

O problema que o algoritmo deve resolver é:

(K-Color):
Instância: Dado um grafo G=(V,E) e um inteiro K, $0 < K < |V|$.
Questão: Existe uma atribuição $c : V \longrightarrow \{1, \ldots, K\}$ tal que $\forall (v, w) \in E$, $c(v) \neq c(w)$?

- Mostre que esse problema pertence ao conjunto NP de problemas.
- Explique porque (K-Color) pertence ao conjunto NP-completo de problemas, que é subconjunto de NP.

2. O algoritmo

O algoritmo deve apresentar uma atribuição que verifica que a instância dada pode ser "colorida" com K cores ou demonstrar que tal atribuição não existe.

Para esta segunda parte é necessário testar todas as possibilidades de atribuir K valores aos vértices do grafo o que, entretanto, não obriga que todas as possíveis atribuições sejam testadas. Veja, por exemplo, o algoritmo de *BackTrack* apresentado no capítulo 9 do livro por Dasgupta, Papadimitiriou e Vazirani, "Algorithms" (citado na ementa do curso).

Um algoritmo deste tipo funciona da seguinte forma: para cada vértice armazena-se o conjunto das cores que este pode receber (inicialmente $\{1, \ldots, K\}$). A cada iteração um vértice recebe uma cor deste conjunto, operação que retira esta cor dos conjuntos de cores possíveis de todos os vértices a ele adjacentes. Esta operação se repete até que todos os vértices tenham recebido uma cor e o algoritmo termina (visto que nesse momento uma atribuição válida foi encontrada), ou até que todos os vértices ainda sem cor atribuída tenham seu conjunto de cores possíveis vazio. Neste, faz-se um *backtrack*. Ou seja, o último vértice que recebeu uma cor troca sua cor por outra do seu conjunto de cores possíveis que ainda não recebeu. Observe que esta troca de cor modifica os conjuntos de cores possíveis de todos os seus vizinhos. Caso todas as cores do seu conjunto tenham sido testadas, faz-se mais um nível de *backtrack* deixando este vértice sem cor atribuída (e atualizando os conjuntos de cores possíveis dos vizinhos) e refazendo esta mesma operação para o vértice que havia sido colorido imediatamente antes.

Observe, que a atribuição de cores é arbitrária, isto é, existem $K!$ atribuições equivalentes. Isto implica que o primeiro vértice a ser colorido pode receber somente a cor 1. A segunda cor a ser usada pode ser a 2 sem precisar testar com outra cor, e assim por diante até a cor K ser usada. Com isso, o número de atribuições testadas é dividido por $K!$.

Sugere-se que esta implementação seja feita utilizando recursão, para reduzir o esforço de codificação.

Finalmente, caso se *aposte* que existe uma atribuição válida, pode-se tentar encontrá-la por métodos heurísticos, isto é, sem garantia de sucesso e que quando não encontram uma atribuição válida, não geram nenhuma informação com relação a sua existência.

Uma heurística simples e frequentemente eficaz é aplicar uma busca em profundidade que atribui aos vértices visitados sempre a menor cor possível. Muitas outras heurísticas (ou meta-heurísticas) menos eficientes mas mais eficazes existem na literatura.

3. Aplicação e Testes

Uma aplicação do algoritmo proposto acima é jogo conhecido como SUDOKU. Neste jogo um mesmo grafo aparece com alguns vértices já coloridos e o objetivo é encontrar uma atribuição de 9 cores aos vértices ainda sem cor atribuída.

Os testes devem ser feitos sobre um conjunto de instâncias de grafos com valores de K indicados e um subconjunto dos vértices com as respectivas cores atribuídas. Em particular haverá uma série de instâncias do SUDOKU. Nestes caso, como sabe-se que existe uma atribuição deve-se explicitar quando foi usada uma heurística e quando não.

As instâncias estão disponíveis na página do curso.